Combining equilibrium logic and dynamic logic

Luis Fariñas del Cerro, Andreas Herzig, Ezgi Iraz Su

▶ **To cite this version:**

Luis Fariñas del Cerro, Andreas Herzig, Ezgi Iraz Su. Combining equilibrium logic and dynamic logic. 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013), Sep 2013, Corunna, Spain. pp.304-316. hal-01228754

HAL Id: hal-01228754
https://hal.science/hal-01228754

Submitted on 13 Nov 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Open Archive TOULOUSE Archive Ouverte (OATAO)

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author-deposited version published in : http://oatao.univ-toulouse.fr/
Eprints ID : 12615

Official URL: http://dx.doi.org/10.1007/978-3-642-40564-8_30

To cite this version : Fariñas del Cerro, Luis and Herzig, Andreas and Su, Ezgi Iraz *Combining equilibrium logic and dynamic logic.* (2013) In: 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013), 15 September 2013 - 19 September 2013 (Corunna, Spain).

Any correspondance concerning this service should be sent to the repository administrator: staff-oatao@listes-diff.inp-toulouse.fr

Combining Equilibrium Logic and Dynamic Logic

Luis Fariñas del Cerro, Andreas Herzig, and Ezgi Iraz Su*

University of Toulouse
IRIT, CNRS
http://www.irit.fr

Abstract. We extend the language of here-and-there logic by two kinds of atomic programs allowing to minimally update the truth value of a propositional variable here or there, if possible. These atomic programs are combined by the usual dynamic logic program connectives. We investigate the mathematical properties of the resulting extension of equilibrium logic: we prove that the problem of logical consequence in equilibrium models is EXPTIME complete by relating equilibrium logic to dynamic logic of propositional assignments.

Keywords: answer-set programming, here-and-there logic, equilibrium logic, propositional dynamic logic, dynamic logic of propositional assignments.

1 Introduction

Answer Set Programming (ASP) is a successful approach in non-monotonic reasoning. Its efficient implementations became a key technology for declarative problem solving in the AI community [7,8]. In recent years many important results have been obtained from a theoretical point of view, such as the definitions of new comprehensive semantics as equilibrium semantics or the proof of important theorems as strong equivalence theorems [14]. These theoretical and practical results show that ASP is central to various approaches in non-monotonic reasoning.

New applications in AI force us to extend the original language of ASP by some new concepts capable of supporting, for example, the representations of modalities, actions, ontologies or updates. Based on a tradition that was started by Alchourrón, Gärdenfors and Makinson and also by Katsuno and Mendelzon [1,13], several researchers have proposed to extend ASP by operations allowing to update or revise a given ASP program through a new piece of information [3,17,15,16]. The resulting formalisms are quite complex, and we think it is fair to say that it is difficult to grasp what the intuitions should be like under these approaches.

We here propose a different, more modest approach, where the new piece of information is restricted to be atomic. It is based on the update of here-and-there (HT) models. Such models are made up of two sets of propositional variables, H ('here') and T ('there'), such that $H \subseteq T$. We consider two kinds of basic update operations: to set a propositional variable true either here or there according to its truth value in these sets;

* We would like to thank the three reviewers of LPNMR 2013 for their helpful comments. This work was partially supported by the French-Spanish *Laboratoire Européen Associé (LEA) "French-Spanish Lab of Advanced Studies in Information Representation and Processing"*.

similarly to set it false either here or there, again if possible. From these basic update operations we allow to build update programs by means of the standard dynamic logic program operators of sequential and nondeterministic composition, iteration, and test. We call the result dynamic here-and-there logic (D-HT).

The notions of an equilibrium model and of logical consequence in equilibrium models can then be defined exactly as before. We show that the problem of satisfiability in HT models and of consequence in equilibrium models are both EXPTIME complete. In order to do so, we use dynamic logic of propositional assignments (DL-PA) that was recently studied in [2]. We define a translation tr_1 from the language of D-HT into the language of DL-PA. Our main result says that a formula φ is an equilibrium consequence of a formula χ if and only if the DL-PA formula

$$\langle \pi_1 \rangle \Big(tr_1(\chi) \wedge {\sim} \langle \pi_2 \rangle tr_1(\chi) \supset tr_1(\varphi) \Big)$$

is valid, where π_1 and π_2 are DL-PA programs whose length is polynomial in the length of χ and φ. This allows to polynomially embed the problems of D-HT satisfiability and consequence in equilibrium models into DL-PA, and so establishes that they are all in EXPTIME. We moreover show that these upper bounds are tight.

The paper is organized as follows. In Section 2 we introduce dynamic here-and-there logic (D-HT) and define consequence in its equilibrium models. In Section 3 we present dynamic logic of propositional assignments (DL-PA) and establish its complexity. In Section 4 we define translations relating the language of D-HT to the language of DL-PA and vice versa. Section 5 concludes.

2 A Dynamic Extension of HT Logic and of Equilibrium Logic

In this section we propose a dynamic extension of the logic of here-and-there (HT), named D-HT. By means of the standard definition of an equilibrium model, that extension also provides a definition of a non-monotonic consequence relation which is a conservative extension of the standard equilibrium consequence relation.

To begin with, we fix a countable set of *propositional variables* ($\mathbb{P}$) whose elements are noted p, q, etc. The language is produced through adding dynamic modalities to the language of HT. The semantics is based on HT models: an HT model is a couple (H, T) such that $H \subseteq T \subseteq \mathbb{P}$. The sets H and T are respectively called 'here' and 'there'. The constraint that $H \subseteq T$ is the so-called heredity constraint of intuitionistic logic. Each of them is a *valuation*, identified with a subset of $\mathbb{P}$. We write $\mathbb{HT}$ for the set of all HT models. So, $\mathbb{HT} = \{(H, T) \ : \ H \subseteq T \subseteq \mathbb{P}\}$.

2.1 The Language $\mathcal{L}_{\text{D-HT}}$

The language $\mathcal{L}_{\text{D-HT}}$ is defined by the following grammar:

$$\varphi ::= p \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid [\pi]\varphi \mid \langle \pi \rangle \varphi$$
$$\pi ::= +p \mid -p \mid \pi ; \pi \mid \pi \cup \pi \mid \pi^* \mid \varphi ?$$

where p ranges over $\mathbb{P}$.

We have only two *atomic programs* in the language, namely $+p$ and $-p$. Each of them minimally updates an HT model, if this is possible: in a sense, the former 'upgrades the truth of p' while the latter 'downgrades the truth of p'. More precisely, the program $+p$ makes p true there, but keeps its truth value same here if p is not included there. However, if p exists there, but not here then it makes p true here while keeping its truth value there; otherwise the program $+p$ fails. On the other hand, the program $-p$ sets p false here as it keeps it there if p is contained here. Nevertheless, if p is only contained there, but not here then the program $-p$ excludes p there keeping its truth value same here; or else the program fails.

The operators of sequential composition (";"), nondeterministic composition ("$\cup$"), finite iteration ("(.)*", the so-called Kleene star), and test ("(.)?") are familiar from propositional dynamic logic (PDL).

An *expression* is a formula or a program.

The *length* of a formula φ, noted $|\varphi|$, is the number of symbols used to write down φ, with the exception of [,], $\langle$, $\rangle$, and parentheses. For example, $|p \wedge (q \vee r)| = 1 + 1 + 3 = 5$. The length of a program π, noted $|\pi|$, is defined in the same way. For example, $|([+p]\bot?\,; -p)| = 4 + 1 + 2 = 7$.

For a given formula φ, the set of variables occurring in φ is noted $\mathbb{P}_\varphi$. For example, $\mathbb{P}_{[-p](q\vee r)} = \{p, q, r\}$.

The *static fragment* of $\mathcal{L}_{\text{D-HT}}$ is the fragment of $\mathcal{L}_{\text{D-HT}}$ without dynamic operators $[\pi]$ and $\langle\pi\rangle$ for every π, noted $\mathcal{L}_{\text{HT}}$. This is nothing but the language of HT and of equilibrium logic.

Negation of a formula φ, noted $\neg\varphi$, is defined as the abbreviation of $\varphi \to \bot$. We also use $\top$ as a shorthand for $\bot \to \bot$.

2.2 Dynamic Here-and-There Logic

We display below the interpretation of formulas and programs together at a time: the interpretation $\|\varphi\|_{\text{D-HT}}$ of a formula φ is a set of HT models, while the interpretation $\|\pi\|_{\text{D-HT}}$ of a program π is a relation on the set of HT models, $\mathbb{HT}$. Note that the interpretation of the dynamic connectives differs from that of usual modal logics because there is a single relation interpreting programs (that therefore does not vary with the models). The definitions are in Table 1.

For instance, $\|\neg p\|_{\text{D-HT}}$ is the set of HT models (H, T) such that $p \notin T$ (and therefore $p \notin H$ by the heredity constraint). Hence, $\|p \vee \neg p\|_{\text{D-HT}}$ is the set of HT models (H, T) such that $p \in H$ or $p \notin T$. $\|\neg\neg p\|_{\text{D-HT}}$ is the set of HT models (H, T) such that $p \in T$. Moreover, $\|\langle+p\rangle\top\|_{\text{D-HT}}$ is the set of HT models (H, T) such that $p \notin H$: when $p \in H$ then p cannot be upgraded and the $+p$ program is inexecutable. Finally, the models of the following formula are all those HT-models (H, T) where T contains p and H does not.

$$\begin{aligned}\|\langle+p\rangle\top \wedge \langle-p\rangle\top\|_{\text{D-HT}} &= \|\neg\neg p\|_{\text{D-HT}} \cap (\mathbb{HT} \setminus \|p\|_{\text{D-HT}}) \\ &= \{(H, T) \;:\; p \notin H \text{ and } p \in T\}\end{aligned}$$

Table 1. Interpretation of the D-HT connectives

$$
\begin{aligned}
\|p\|_{\text{D-HT}} &= \{(H,T) \ : \ p \in H\} \\
\|\bot\|_{\text{D-HT}} &= \emptyset \\
\|\varphi \wedge \psi\|_{\text{D-HT}} &= \|\varphi\|_{\text{D-HT}} \cap \|\psi\|_{\text{D-HT}} \\
\|\varphi \vee \psi\|_{\text{D-HT}} &= \|\varphi\|_{\text{D-HT}} \cup \|\psi\|_{\text{D-HT}} \\
\|\varphi \to \psi\|_{\text{D-HT}} &= \{(H,T) \ : \ (H,T),(T,T) \in (\mathbb{HT} \setminus \|\varphi\|_{\text{D-HT}}) \cup \|\psi\|_{\text{D-HT}}\} \\
\|[\pi]\varphi\|_{\text{D-HT}} &= \{(H,T) \ : \ (H_1,T_1) \in \|\varphi\|_{\text{D-HT}} \text{ for every } ((H,T),(H_1,T_1)) \in \|\pi\|_{\text{D-HT}}\} \\
\|\langle\pi\rangle\varphi\|_{\text{D-HT}} &= \{(H,T) \ : \ (H_1,T_1) \in \|\varphi\|_{\text{D-HT}} \text{ for some } ((H,T),(H_1,T_1)) \in \|\pi\|_{\text{D-HT}}\} \\
\|{+p}\|_{\text{D-HT}} &= \{((H_1,T_1),(H_2,T_2)) \ : \ H_2 \setminus H_1 = \{p\} \text{ and } T_2 = T_1, \text{ or } T_2 \setminus T_1 = \{p\} \text{ and } H_2 = H_1\} \\
\|{-p}\|_{\text{D-HT}} &= \{((H_1,T_1),(H_2,T_2)) \ : \ H_1 \setminus H_2 = \{p\} \text{ and } T_2 = T_1, \text{ or } T_1 \setminus T_2 = \{p\} \text{ and } H_2 = H_1\} \\
\|\pi_1;\pi_2\|_{\text{D-HT}} &= \|\pi_1\|_{\text{D-HT}} \circ \|\pi_2\|_{\text{D-HT}} \\
\|\pi_1 \cup \pi_2\|_{\text{D-HT}} &= \|\pi_1\|_{\text{D-HT}} \cup \|\pi_2\|_{\text{D-HT}} \\
\|\pi^*\|_{\text{D-HT}} &= (\|\pi\|_{\text{D-HT}})^* \\
\|\varphi?\|_{\text{D-HT}} &= \{((H,T),(H,T)) \ : \ (H,T) \in \|\varphi\|_{\text{D-HT}}\}
\end{aligned}
$$

A formula φ is D-HT *valid* if and only if every HT model is also a model of φ, i.e., $\|\varphi\|_{\text{D-HT}} = \mathbb{HT}$. For example, neither $\langle{+p}\rangle\top$ nor $\langle{-p}\rangle\top$ is valid, but $\langle{+p} \cup {-p}\rangle\top$ is. Moreover, $[+p][+p]p$, $[-p][-p]\neg p$, and $[p? \cup \neg p?](p \vee \neg p)$ are all valid. Finally, the following equivalences are valid:

$$
\begin{aligned}
[-p]\bot &\leftrightarrow \neg p \\
\langle -p\rangle\top &\leftrightarrow \neg\neg p \\
[+p]\bot &\leftrightarrow p
\end{aligned}
$$

Therefore $[-p]\bot$, $\langle -p\rangle\top$ and $[+p]\bot$ can all be expressed in $\mathcal{L}_{HT}$. In contrast, $\langle +p\rangle\top$ cannot because there is no formula in the static fragment $\mathcal{L}_{\text{HT}}$ that conveys that $p \in T \setminus H$. So our extension of HT is more expressive than HT itself.

D-HT logic satisfies the heredity property of intuitionistic logic for atomic formulas: if (H,T) is an HT model of p then (T,T) is also an HT model of p. It is trivially satisfied because for every HT model (H,T), $H \subseteq T$. D-HT logic however fails to satisfy that property for more complex formulas containing dynamic operators. To see this, consider the HT model $(\emptyset,\{p\})$ and the formula $\langle +p\rangle\top$: $(\emptyset,\{p\})$ is a model of $\langle +p\rangle\top$, while $(\{p\},\{p\})$ is not.

Our logic D-HT is a particular intuitionistic modal logic. Such logics were studied in the literature [6]. For such logics, duality of the modal operators fails: while $[\pi]\varphi \to \neg\langle\pi\rangle\neg\varphi$ is valid, the converse is invalid. For example, $(\emptyset,\emptyset)$ is an HT model of $\neg\langle +p\rangle\neg p$, but not of $[+p]p$.

It follows from the next proposition that we have a finite model property for D-HT: if φ has an HT model then φ has an HT model (H,T) such that T is finite.

Proposition 1. *Let φ be an $\mathcal{L}_{\text{D-HT}}$ formula. Let P be a set of propositional variables such that $P \cap \mathbb{P}_\varphi = \emptyset$, and let $Q \subseteq P$. Then, $(H,T) \in \|\varphi\|_{\text{D-HT}}$ iff $(H \cup Q, T \cup P) \in \|\varphi\|_{\text{D-HT}}$.*

2.3 Dynamic Equilibrium Logic

An *equilibrium model* of an $\mathcal{L}_{\mathsf{D\text{-}HT}}$ formula φ is a set of propositional variables $T \subseteq \mathbb{P}$ such that:

1. (T, T) is an HT model of φ;
2. no (H, T) with $H \subset T$ is an HT model of φ.

The valid formulas of D-HT all have exactly one equilibrium model, viz. the empty set. There are formulas that have no equilibrium model, such as $\neg\neg p$. The equilibrium models of equivalent formulas $p \vee \neg p$ and $\neg\neg p \to p$ are $\emptyset$ and $\{p\}$. The only equilibrium model of $\neg p \to q$ is $\{q\}$, and of $\langle +p\rangle(\neg p \to q)$ is $\emptyset$. $\{p\}$ is the only equilibrium model for both $\langle -p\rangle(\neg p \to q)$, and $\langle +q; +q\rangle(p \wedge q)$. However, $\langle -q\rangle(p \wedge q)$ has no equilibrium model because $\langle -q\rangle(p \wedge q)$ does not even have a D-HT model either.

Let χ and φ be $\mathcal{L}_{\mathsf{D\text{-}HT}}$ formulas. φ is a *consequence of* χ *in equilibrium models*, written $\chi \mathrel{|\!\approx} \varphi$, if and only if for every equilibrium model T of χ, (T, T) is an HT model of φ. For example, $\top \mathrel{|\!\approx} \neg p$, $p \vee q \mathrel{|\!\approx} [\neg p?]q$, and $p \vee q \mathrel{|\!\approx} [\neg p?]\langle +p; +p\rangle(p \wedge q)$.

In our dynamic language we can check not only problems of the form $\chi \mathrel{|\!\approx} [\pi]\varphi$, but also problems of the form $\langle \pi\rangle\chi \mathrel{|\!\approx} \varphi$. The former expresses a hypothetical update of χ: if χ is updated by π then φ follows. The latter may express an actual update of χ, where the program π executes the update 'the other way round': it is the converse of the original update program. For example, suppose we want to update $\chi = p \wedge q$ by $\neg q$. Updates by the latter formula can be implemented by the program $-q; -q$. Now the converse execution of $-q; -q$ is nothing but the execution of the program $\pi = +q; +q$. Therefore, in order to know whether the update of $p \wedge q$ by $\neg q$ results in $p \wedge \neg q$ we have to check whether $\langle +q; +q\rangle(p \wedge q) \mathrel{|\!\approx} p \wedge \neg q$. The latter is indeed the case: we have seen above that the only equilibrium model of $\langle +q; +q\rangle(p \wedge q)$ is $\{p\}$, and $(\{p\}, \{p\})$ is clearly a D-HT model of $p \wedge \neg q$.

3 DL-PA: Dynamic Logic of Propositional Assignments

In this section we define syntax and semantics of dynamic logic of propositional assignments (DL-PA) and state complexity results. The star-free fragment of DL-PA was introduced in [9], where it was shown that it embeds Coalition Logic of Propositional Control [10,11,12]. The full logic with the Kleene star was further studied in [2]. In addition to assignments of propositional variables to true or false, here we allow of assignments to arbitrary formulas as well. We need this extension for the purpose of copying the propositional variables of a valuation and similarly, after some changes to be able to retrieve the initial truth values of that valuation. We will explain these notions later in full detail. However, we keep on calling that logic DL-PA. This is in order because it has the same expressivity and the same complexity as the logic DL-PA of [2].

3.1 Language

The language of DL-PA is defined by the following grammar:

$$\pi ::= p{:=}\varphi \mid \pi;\pi \mid \pi \cup \pi \mid \pi^* \mid \varphi?$$
$$\varphi ::= p \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \supset \varphi \mid \langle\pi\rangle\varphi$$

where p ranges over a fixed set of propositional variables $\mathbb{P}$. So, an atomic program of the language of DL-PA is a program of the form $p{:=}\varphi$. The program operators of sequential composition (";"), nondeterministic composition ("$\cup$"), finite iteration ("$(.)^*$"), and test ("$(.)?$") are familiar from Propositional Dynamic Logic (PDL).

The *star-free fragment* of DL-PA is the subset of the language made up of formulas without the Kleene star "$(.)^*$".

We abbreviate the other logical connectives in the usual way; for example, $\sim\varphi$ is defined as $\varphi \supset \bot$. In particular, $\top$ is defined as $\sim\bot = \bot\supset\bot$. Moreover, $[\pi]\varphi$ abbreviates $\sim\langle\pi\rangle\sim\varphi$. The program `skip` abbreviates $\top?$ ("nothing happens"). (Note that it could also be defined by $p{:=}p$, for arbitrary p.) The language of DL-PA allows to express the primitives of standard programming languages. For example, the loop "while φ do π" can be expressed as the DL-PA program $(\varphi?;\pi)^*;\sim\varphi?$.

3.2 Semantics

DL-PA programs are interpreted by means of a (unique) *relation between valuations*: atomic programs $p{:=}\varphi$ update valuations in the obvious way, and complex programs are interpreted just as in PDL by mutual recursion. Table 2 gives the interpretation of the DL-PA connectives.

Table 2. Interpretation of the DL-PA connectives

$$\begin{aligned}
||p{:=}\varphi||_{\text{DL-PA}} &= \{(V,V') : \text{if } V \in ||\varphi||_{\text{DL-PA}} \text{ then } V' = V\cup\{p\} \text{ and if } V \notin ||\varphi||_{\text{DL-PA}} \text{ then } V' = V\setminus\{p\}\}\\
||\pi;\pi'||_{\text{DL-PA}} &= ||\pi||_{\text{DL-PA}} \circ ||\pi'||_{\text{DL-PA}}\\
||\pi\cup\pi'||_{\text{DL-PA}} &= ||\pi||_{\text{DL-PA}} \cup ||\pi'||_{\text{DL-PA}}\\
||\pi^*||_{\text{DL-PA}} &= (||\pi||_{\text{D-HT}})^*\\
||\varphi?||_{\text{DL-PA}} &= \{(V,V) \;:\; V \in ||\varphi||_{\text{DL-PA}}\}
\end{aligned}$$

$$\begin{aligned}
||p||_{\text{DL-PA}} &= \{V \;:\; p \in V\}\\
||\bot||_{\text{DL-PA}} &= \emptyset\\
||\varphi\wedge\psi||_{\text{DL-PA}} &= ||\varphi||_{\text{DL-PA}} \cap ||\psi||_{\text{DL-PA}}\\
||\varphi\vee\psi||_{\text{DL-PA}} &= ||\varphi||_{\text{DL-PA}} \cup ||\psi||_{\text{DL-PA}}\\
||\varphi\supset\psi||_{\text{DL-PA}} &= (2^{\mathbb{P}} \setminus ||\varphi||_{\text{DL-PA}}) \cup ||\psi||_{\text{DL-PA}}\\
||\langle\pi\rangle\varphi||_{\text{DL-PA}} &= \{V \;:\; \text{there is } V' \text{ such that } (V,V') \in ||\pi||_{\text{DL-PA}} \text{ and } V' \in ||\varphi||_{\text{DL-PA}}\}
\end{aligned}$$

A formula φ is DL-PA *valid* if $||\varphi||_{\text{DL-PA}} = 2^{\mathbb{P}}$, and it is DL-PA *satisfiable* if $||\varphi||_{\text{DL-PA}} \neq \emptyset$. For example, the formulas $\langle p{:=}\top\rangle\top$, $\langle p{:=}\top\rangle p$ and $\langle p{:=}\bot\rangle\sim p$ are all valid, as well as $\psi \wedge [\psi?]\varphi \supset \varphi$ and $[p{:=}\top \cup q{:=}\top](p \vee q)$. Moreover, if p does not occur in φ then both $\varphi \supset \langle p{:=}\top\rangle\varphi$ and $\varphi \supset \langle p{:=}\bot\rangle\varphi$ are valid. This is due to the following property that we will use while translating dynamic equilibrium logic into DL-PA.

Proposition 2. *Suppose $\mathbb{P}_\varphi \cap P = \emptyset$, i.e., none of the variables of P occurs in φ. Then $V \cup P \in \|\varphi\|_{\mathsf{DL\text{-}PA}}$ iff $V \setminus P \in \|\varphi\|_{\mathsf{DL\text{-}PA}}$.*

Contrarily to PDL, it is shown in [2] that the Kleene star operator can be eliminated in DL-PA: for every DL-PA program π, there is an equivalent program π' such that no Kleene star occurs in π'. However, the elimination is not polynomial.

3.3 Complexity of the Full Language

It is proved in [2] that both model and satisfiability checking are EXPTIME complete for the fragment of DL-PA including the conversion operator and restricting the formulas φ in atomic programs $p{:=}\varphi$ to either $\top$ or $\bot$. The lower bounds for both problems clearly transfer.

The upper bound for the satisfiability problem is established in [2] by means of a polynomial transformation into the satisfiability problem of PDL. An inspection of the proof shows that it generalizes to arbitrary assignments. So, the satisfiability problem of our DL-PA has the same complexity as that of PDL: it is EXPTIME complete.

The upper bound for the model checking problem can be established just as in [2] by polynomially transforming it into the satisfiability problem: we use that $V \in \|\varphi\|_{\mathsf{DL\text{-}PA}}$ if and only if the formula $\varphi \wedge (\bigwedge_{p \in V} p) \wedge (\bigwedge_{p \notin V} {\sim}p)$ is satisfiable. So, the model checking problem of our DL-PA is EXPTIME complete, too.

3.4 Complexity of the Star-Free Fragment

The complexity of the decision problems for the star-free fragment of the language of [2] is established in [9], where it is shown that it is PSPACE complete for both model and satisfiability checking.

As to model checking, the lower bound clearly transfers to our star-free fragment. Furthermore, the PSPACE model checking algorithm of [9] can be extended to our more general star-free fragment without conversion and with general assignment $p{:=}\varphi$.

As to satisfiability checking, the lower bound of [9] transfers. The upper bound can be proved in the same way as in [9]: given a formula φ, nondeterministically guess a valuation V and model check whether $V \in \|\varphi\|_{\mathsf{DL\text{-}PA}}$. Model checking being in PSPACE, satisfiability checking must therefore be in NPSPACE, and NPSPACE is the same complexity class as PSPACE due to Savitch's theorem.

4 Relating D-HT and DL-PA

In this section we are going to translate D-HT and dynamic equilibrium logic into DL-PA, and vice versa. The translation is polynomial and allows to check D-HT validity and consequence in equilibrium models. This establishes an EXPTIME upper bound for the complexity of the latter problem. We also show that the upper bound is tight.

We start by defining some DL-PA programs that will be the building blocks in embedding some notions of D-HT into DL-PA. Some of these programs require to copy propositional variables.

4.1 Copying Propositional Variables

The translation introduces fresh propositional variables that do not not exist in the formula we translate. Precisely, this requires to suppose a new set of propositional variables: it is the union of the set of 'original' variables $\mathbb{P} = \{p_1, p_2, \ldots\}$ and the set of 'copies' of these variables $\mathbb{P}' = \{p'_1, p'_2, \ldots\}$, where $\mathbb{P}$ and $\mathbb{P}'$ are disjoint. The function $(.)'$ is a bijection between these two sets: for every subset $Q \subseteq \mathbb{P}$ of original variables, the set $Q' = \{p' \ : \ p \in Q\} \subseteq \mathbb{P}'$ is its image, and the other way around. We suppose that $(.)'$ is an involution, i.e., it behaves as an identity when applied twice. Now, a DL-PA valuation extends to the form of $X \cup Y'$, where $X \subseteq \mathbb{P}$ and $Y' \subseteq \mathbb{P}'$. As a result, DL-PA validity expands to the power set of $\mathbb{P} \cup \mathbb{P}'$, i.e., $2^{\mathbb{P} \cup \mathbb{P}'}$. In our embedding, X will encode the here-valuation and Y' will encode the there-valuation. Note that in order to respect the heredity constraint hidden in the structure of here-and there models, our translation has to guarantee that X is a subset of Y.

4.2 Useful DL-PA Programs

Table 3 collects some DL-PA programs that are going to be convenient for our enterprise. In that table, $\{p_1, \ldots, p_n\}$ is some finite subset of $\mathbb{P}$ and each p'_i is a copy of p_i as explained above. For $n = 0$ we stipulate that all these programs equal `skip`.

Table 3. Some useful DL-PA programs

$$\texttt{mkFalse}^{\geq 0}(\{p_1, \ldots, p_n\}) = (p_1{:=}\bot \cup \texttt{skip});\ \cdots\ ;(p_n{:=}\bot \cup \texttt{skip})$$
$$\texttt{mkFalse}^{>0}(\{p_1, \ldots, p_n\}) = (p_1{:=}\bot \cup \cdots \cup p_n{:=}\bot); \texttt{mkFalse}^{\geq 0}(P)$$
$$\texttt{cp}(\{\texttt{p}_1, \ldots, \texttt{p}_\texttt{n}\}) = p'_1{:=}p_1;\ \cdots\ ; p'_n{:=}p_n$$
$$\texttt{cpBack}(\{\texttt{p}_1, \ldots, \texttt{p}_\texttt{n}\}) = p_1{:=}p'_1;\ \cdots\ ; p_n{:=}p'_n$$

Let $P = \{p_1, \ldots, p_n\}$. The program $\texttt{mkFalse}^{\geq 0}(P)$ nondeterministically makes some of the variables of P false, possibly none. The program $\texttt{mkFalse}^{>0}(P)$ nondeterministically makes false at least one of the variables of P, and possibly more. Its subprogram $p_1{:=}\bot \cup \cdots \cup p_n{:=}\bot$ makes exactly one of the variables in the valuation P false. The program $\texttt{cp}(\texttt{P})$ assigns to each 'fresh' variable p'_i the truth value of p_i, while the program $\texttt{cpBack}(\texttt{P})$ assigns to each variable p_i the truth value of p'_i. We shall use the former as a way of storing the truth value of each variable of P before they undergo some changes. That will allow later on to retrieve the original values of the variables in P by means of the $\texttt{cpBack}(\texttt{P})$ program. Therefore the sequence $\texttt{cp}(\texttt{P}); \texttt{cpBack}(\texttt{P})$ leaves the variables in P unchanged.

Observe that each program of Table 3 has length linear in the cardinality of P. Observe also that the programs $\texttt{mkFalse}^{\geq 0}(P)$ and $\texttt{mkFalse}^{>0}(P)$ are nondeterministic. In contrast, the programs $\texttt{cp}(\texttt{P})$ and $\texttt{cpBack}(\texttt{P})$ are deterministic and always executable: $[\texttt{cp}(\texttt{P})]\varphi$ and $\langle \texttt{cp}(\texttt{P}) \rangle \varphi$ are equivalent, as well as $[\texttt{cpBack}(\texttt{P})]\varphi$ and $\langle \texttt{cpBack}(\texttt{P}) \rangle \varphi$.

Lemma 1 (Program Lemma). *Let $P \subseteq \mathbb{P}$ be finite and non-empty. Then*

$$\begin{aligned}
\|\mathtt{mkFalse}^{\geq 0}(P)\|_{\mathsf{DL\text{-}PA}} &= \{(V_1, V_2) \ : \ V_2 = V_1 \setminus Q, \textit{ for some } Q \subseteq P\} \\
\|\mathtt{mkFalse}^{>0}(P)\|_{\mathsf{DL\text{-}PA}} &= \{(V_1, V_2) \ : \ V_2 = V_1 \setminus Q, \textit{ for some } Q \subseteq P \textit{ such that } Q \neq \emptyset\} \\
\|\mathtt{cp(P)}\|_{\mathsf{DL\text{-}PA}} &= \{(X_1 \cup Y_1', X_2 \cup Y_2') \ : \ X_2 = X_1 \textit{ and } Y_2' = (X_1 \cap P)' \cup (Y_1' \setminus P')\} \\
\|\mathtt{cpBack(P)}\|_{\mathsf{DL\text{-}PA}} &= \{(X_1 \cup Y_1', X_2 \cup Y_2') \ : \ X_2 = (Y_1' \cap P')' \cup (X_1 \setminus P) \textit{ and } Y_2' = Y_1'\}.
\end{aligned}$$

It follows from the interpretations of $\mathtt{cp(P)}$ and $\mathtt{mkFalse}^{\geq 0}(P)$ that

$$\begin{aligned}
\|\mathtt{cp(P)}; \mathtt{mkFalse}^{\geq 0}(P)\|_{\mathsf{DL\text{-}PA}} = \{(X_1 \cup Y_1', X_2 \cup Y_2') \ : X_2 = X_1 \setminus Q \text{ for some } Q \subseteq P \\
\text{and } Y_2' = (X_1 \cap P)' \cup (Y_1' \setminus P')\}.
\end{aligned}$$

4.3 Translating $\mathcal{L}_{\mathsf{D\text{-}HT}}$ to $\mathcal{L}_{\mathsf{DL\text{-}PA}}$

To start with we translate the formulas and programs of the language $\mathcal{L}_{\mathsf{D\text{-}HT}}$ into the language $\mathcal{L}_{\mathsf{DL\text{-}PA}}$. The translation is given in Table 4 in terms of a recursively defined mapping tr_1, where we have omitted the homomorphic cases such as $tr_1([\pi]\varphi) = [tr_1(\pi)]tr_1(\varphi)$ and $tr_1(\varphi?) = (tr_1(\varphi))?$.

Table 4. Translation from $\mathcal{L}_{\mathsf{D\text{-}HT}}$ into DL-PA

$$\begin{aligned}
tr_1(p) &= p, \quad \text{for } p \in \mathbb{P} \\
tr_1(\varphi \to \psi) &= [\mathtt{skip} \cup \mathtt{cpBack}(\mathbb{P}_{\varphi \to \psi})](tr_1(\varphi) \supset tr_1(\psi)) \\
tr_1(+p) &= (\sim p'? \,;\, p' := \top) \cup (\sim p \wedge p'? \,;\, p := \top) \\
tr_1(-p) &= (p? \,;\, p := \bot) \cup (\sim p \wedge p'? \,;\, p' := \bot)
\end{aligned}$$

Observe that tr_1 is polynomial. For example,

$$\begin{aligned}
tr_1(\top) &= tr_1(\bot \to \bot) = [\mathtt{skip} \cup \mathtt{skip}]\top \\
tr_1(p \vee \neg p) &= p \vee [\mathtt{skip} \cup p := p']{\sim}p \\
tr_1(p \to q) &= [\mathtt{skip} \cup (p := p' \,;\, q := q')](p \supset q)
\end{aligned}$$

The first formula is equivalent to $\top$. The second is equivalent to $p \vee (\sim p \wedge \sim p')$, i.e., to $p \vee \sim p'$. The third is equivalent to $(p \supset q) \wedge (p' \supset q')$.

Lemma 2 (Main Lemma). $(H, T) \in \|\varphi\|_{\mathsf{D\text{-}HT}}$ *if and only if* $H \cup T' \in \|tr_1(\varphi)\|_{\mathsf{DL\text{-}PA}}$.

Proof is by induction on the length of expressions (formulas or programs): we show that for every expression ξ,

- if ξ is a formula then $(H, T) \in \|\xi\|_{\mathsf{D\text{-}HT}}$ if and only if $H \cup T' \in \|tr_1(\xi)\|_{\mathsf{DL\text{-}PA}}$, and
- if ξ is a program then
  $((H_1, T_1), (H_2, T_2)) \in \|\xi\|_{\mathsf{D\text{-}HT}}$ if and only if $((H_1 \cup T_1'), (H_2 \cup T_2')) \in \|tr_1(\xi)\|_{\mathsf{DL\text{-}PA}}$.

4.4 From D-HT to DL-PA

We now establish how tr_1 can be used to prove that a given formula φ is D-HT satisfiable. To that end, we prefix the translation by the '$\mathtt{cp}(\mathbb{P}_\varphi)$' program that is followed by the '$\mathtt{mkFalse}^{\geq 0}(\mathbb{P}_\varphi)$' program. The '$\mathtt{cp}(\mathbb{P}_\varphi)$' program produces a 'classical' valuation $T \cup T'$, for some subset T of $\mathbb{P}$ (as far as the variables of φ are concerned), and then '$\mathtt{mkFalse}^{\geq 0}(\mathbb{P}_\varphi)$' program transforms the valuation $T \cup T'$ into a valuation $H \cup T'$ for some H such that $H \subseteq T$.

Theorem 1. *Let φ be an $\mathcal{L}_{\mathsf{D\text{-}HT}}$ formula. Then*

- *φ is* D-HT *satisfiable iff $\langle \mathtt{cp}(\mathbb{P}_\varphi)\rangle\langle \mathtt{mkFalse}^{\geq 0}(\mathbb{P}_\varphi)\rangle tr_1(\varphi)$ is* DL-PA *satisfiable, and*
- *φ is* D-HT *valid iff $[\mathtt{cp}(\mathbb{P}_\varphi)][\mathtt{mkFalse}^{\geq 0}(\mathbb{P}_\varphi)]tr_1(\varphi)$ is* DL-PA *valid.*

This is proved by the Main Lemma and the Program Lemma.

As a result of the theorem above, the formula $[\mathtt{cp}(\{\mathtt{p}\})][\mathtt{mkFalse}^{\geq 0}(\{p\})]tr_1(p \vee \neg p)$ should not be DL-PA valid since we know that $p \vee \neg p$ is not D-HT valid. We indeed have the following sequence of equivalent formulas:

1. $[\mathtt{cp}(\{\mathtt{p}\})][\mathtt{mkFalse}^{\geq 0}(\{p\})]tr_1(p \vee \neg p)$
2. $[p'{:=}p][p{:=}\bot \cup \mathtt{skip}](p \vee [\mathtt{skip} \cup p{:=}p']{\sim}p)$
3. $[p'{:=}p][p{:=}\bot \cup \mathtt{skip}](p \vee {\sim}p')$
4. $[p'{:=}p]([p{:=}\bot](p \vee {\sim}p') \wedge (p \vee {\sim}p'))$
5. $[p'{:=}p]({\sim}p' \wedge (p \vee {\sim}p'))$
6. ${\sim}p \wedge (p \vee {\sim}p)$
7. ${\sim}p$

The last is obviously not DL-PA valid, so the first line is not DL-PA valid either.

4.5 From Dynamic Equilibrium Logic to DL-PA

Having seen how D-HT can be embedded into DL-PA, we now turn to equilibrium logic.

Theorem 2. *For every $\mathcal{L}_{\mathsf{D\text{-}HT}}$ formula χ, $T \subseteq \mathbb{P}$ is an equilibrium model of χ if and only if $T \cup T'$ is a* DL-PA *model of $tr_1(\chi) \wedge {\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle tr_1(\chi)$.*

Proof. $T \cup T'$ is a DL-PA model of $tr_1(\chi) \wedge {\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle tr_1(\chi)$ if and only if

$$T \cup T' \text{ is a DL-PA model of } tr_1(\chi) \tag{1}$$

and

$$T \cup T' \text{ is a DL-PA model of } {\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle tr_1(\chi) \tag{2}$$

By the Main Lemma, (1) is the case if and only if (T, T) is a HT model of χ in D-HT. It remains to prove that (2) is the case if and only if (H, T) is not a HT model of χ, for any set $H \subset T$. We establish this by proving that the following statements are equivalent.

1. $T \cup T'$ is a DL-PA model of ${\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle tr_1(\chi)$
2. $(T \cap \mathbb{P}_\chi) \cup T'$ is not a DL-PA model of $\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle tr_1(\chi)$ (Proposition 2)

3. $H \cup T'$ is not a DL-PA model of $tr_1(\chi)$, for any $H \subset T \cap \mathbb{P}_\chi$ (Program Lemma 1)
4. H, T is not a HT model of χ, for any set $H \subset T \cap \mathbb{P}_\chi$ (Main Lemma 2)
5. H, T is not a HT model of χ, for any set $H \subset T$ (Proposition 1).

q.e.d.

Theorem 3. *Let χ and φ be $\mathcal{L}_{\text{D-HT}}$ formulas. Then $\chi \approx \varphi$ if and only if*

$$\langle \mathsf{cp}(\mathbb{P}_\chi \cup \mathbb{P}_\varphi)\rangle\Big((tr_1(\chi) \wedge {\sim}\langle \mathtt{mkFalse}^{>0}(\mathbb{P}_\chi)\rangle tr_1(\chi)) \supset tr_1(\varphi)\Big)$$

is DL-PA *valid.*

Theorem 3 provides a polynomial embedding of the consequence problem in our dynamic equilibrium logic into DL-PA. Together with the EXPTIME upper bound for the validity problem of DL-PA that we have established in Section 3.3, it follows that the former problem is in EXPTIME. In the next section we establish that the upper bound is tight.

4.6 From DL-PA to D-HT

We establish EXPTIME hardness of the D-HT satisfiability problem by means of a simple translation of the fragment of DL-PA whose atomic assignment programs are restricted to $p{:=}\top$ and $p{:=}\bot$ and with the conversion operator: the result follows because it is known that the satisfiability problem for that fragment is already EXPTIME hard [2].

The translation is given in Table 5, where we have omitted the homomorphic cases. In the last two lines, $tr_2(p{:=}\top)$ makes p true both here and there, while $tr_2(p{:=}\bot)$ makes p false both here and there. The translation is clearly polynomial.

Table 5. Translation from DL-PA with assignments only to $\top$ and $\bot$ into $\mathcal{L}_{\text{D-HT}}$ (main cases)

$$\begin{aligned} tr_2(p) &= p, \qquad \text{for } p \in \mathbb{P} \\ tr_2(\varphi \supset \psi) &= tr_2(\varphi) \rightarrow tr_2(\psi) \\ tr_2(p{:=}\top) &= p? \cup (+p\,;+p) \\ tr_2(p{:=}\bot) &= \neg p? \cup (-p\,;-p) \end{aligned}$$

The next lemma is the analog of the Main Lemma adapted to tr_2, and is used in the proof of Theorem 4.

Lemma 3. *Let φ be a* DL-PA *formula. Then,*

$$V \in \|\varphi\|_{\text{DL-PA}} \text{ if and only if } (V, V) \in \|tr_2(\varphi)\|_{\text{D-HT}}.$$

Now, we are ready to show how tr_2 can be used to prove that a given formula φ is DL-PA satisfiable.

Theorem 4. *Let φ be a* DL-PA *formula. Then, φ is* DL-PA *satisfiable if and only if $tr_2(\varphi) \wedge \bigwedge_{p \in \mathbb{P}_\varphi}(p \vee \neg p)$ is satisfiable in* D-HT.

Since the satisfiability problem for the fragment of the language of DL-PA with assignments only to $\top$ and $\bot$ is EXPTIME hard [2], through the theorem above we deduce that the D-HT validity problem is also EXPTIME hard; moreover, Theorem 1 tells us that it is actually EXPTIME complete.

The complexity of the equilibrium consequence problem is at least that of the validity problem in D-HT. Therefore, the consequence problem in dynamic equilibrium logic is EXPTIME hard, too. Moreover, Theorem 3 tells us that it is actually EXPTIME complete.

5 Conclusion

We have defined a simple logic D-HT of atomic change of equilibrium models and have shown that it is strongly related to dynamic logic of propositional assignments (DL-PA). This in particular allows to obtain EXPTIME complexity results both for the D-HT satisfiability and for the consequence in its equilibrium models.

The present paper is part of a line of work aiming at reexamining the logical foundations of equilibrium logic and ASP. In previous works we had analyzed equilibrium logic by means of the concepts of contingency [4] and by means of modal operators quantifying over here-and -there worlds in the definition of an equilibrium model [5]. The present paper adds an analysis of the dynamics by integrating operators of upgrading and downgrading propositional variables.

What about updates by complex programs? Actually we may implement such updates by means of complex D-HT programs. For example, the D-HT program

$$(\neg p \vee q)? \cup (-p; +q)$$

makes the implication $p \rightarrow q$ true, whatever the initial HT model is (although there may be other minimal ways of achieving this). More generally, let us consider that an abstract semantical update operation is a function $f : \mathbb{HT} \longrightarrow 2^{\mathbb{HT}}$ associating to every HT model (H, T) the set of HT models $f(H, T)$ resulting from the update. If the language is finite then for every such f we can design a program π_f such that $\|\pi_f\|_{\text{D-HT}} = f$, viz. the graph of f. This makes use of the fact that in particular we can uniquely (up to logical equivalence) characterize HT models by means of the corresponding formulas. For example, the formula $(\langle +p\rangle\top \wedge \langle -p\rangle\top) \wedge (\bigwedge_{q\neq p} \neg q)$ identifies the HT model $(\emptyset, \{p\})$. Note finally that we cannot express the HT model $(\emptyset, \{p\})$ in the language $\mathcal{L}_{\text{HT}}$, where there is no formula distinguishing that model from the model $(\{p\}, \{p\})$.

References

1. Alchourrón, C., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. J. of Symbolic Logic 50, 510–530 (1985)
2. Balbiani, P., Herzig, A., Troquard, N.: Dynamic logic of propositional assignments: a well-behaved variant of PDL. In: Kupferman, O. (ed.) Logic in Computer Science (LICS), New Orleans, June 25-28. IEEE (2013), http://www.ieee.org/
3. Eiter, T., Fink, M., Sabbatini, G., Tompits, H.: Using methods of declarative logic programming for intelligent information agents. TPLP 2(6), 645–709 (2002)

4. Fariñas del Cerro, L., Herzig, A.: Contingency-based equilibrium logic. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS, vol. 6645, pp. 223–228. Springer, Heidelberg (2011), http://www.springerlink.com
5. Fariñas del Cerro, L., Herzig, A.: The modal logic of equilibrium models. In: Tinelli, C., Sofronie-Stokkermans, V. (eds.) FroCoS 2011. LNCS, vol. 6989, pp. 135–146. Springer, Heidelberg (2011), http://www.springerlink.com
6. Fischer-Servi, G.: On modal logic with an intuitionistic base. Studia Logica 36(4), 141–149 (1976)
7. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Conflict-driven answer set solving. In: Veloso, M.M. (ed.) IJCAI, pp. 386–392 (2007)
8. Gebser, M., Ostrowski, M., Schaub, T.: Constraint answer set solving. In: Hill, P.M., Warren, D.S. (eds.) ICLP 2009. LNCS, vol. 5649, pp. 235–249. Springer, Heidelberg (2009)
9. Herzig, A., Lorini, E., Moisan, F., Troquard, N.: A dynamic logic of normative systems. In: Walsh, T. (ed.) International Joint Conference on Artificial Intelligence (IJCAI), Barcelona, pp. 228–233. IJCAI/AAAI (2011), Erratum at http://www.irit.fr/~Andreas.Herzig/P/Ijcai11.html
10. van der Hoek, W., Walther, D., Wooldridge, M.: On the logic of cooperation and the transfer of control. J. of AI Research (JAIR) 37, 437–477 (2010)
11. van der Hoek, W., Wooldridge, M.: On the dynamics of delegation, cooperation and control: a logical account. In: Proc. AAMAS 2005 (2005)
12. van der Hoek, W., Wooldridge, M.: On the logic of cooperation and propositional control. Artif. Intell. 164(1-2), 81–119 (2005)
13. Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. In: Gärdenfors, P. (ed.) Belief Revision, pp. 183–203. Cambridge University Press (1992); preliminary version in Allen, J.A., Fikes, R., and Sandewall, E. (eds.) Principles of Knowledge Representation and Reasoning: Proc. 2nd Int. Conf., pp. 387–394. Morgan Kaufmann Publishers (1991)
14. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic 2(4), 526–541 (2001)
15. Slota, M., Leite, J.: Robust equivalence models for semantic updates of answer-set programs. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) KR. AAAI Press (2012)
16. Slota, M., Leite, J.: A unifying perspective on knowledge updates. In: del Cerro, L.F., Herzig, A., Mengin, J. (eds.) JELIA 2012. LNCS, vol. 7519, pp. 372–384. Springer, Heidelberg (2012)
17. Zhang, Y., Foo, N.Y.: A unified framework for representing logic program updates. In: Veloso, M.M., Kambhampati, S. (eds.) AAAI, pp. 707–713. AAAI Press/The MIT Press (2005)